Authorship Compilation

Allyson Park

Discussion questions written for UC Berkeley's CS61B: Data Structures + Algorithms

Contents

1 Senior Class

For each line in the `main` method of our `testPeople` class, if something is printed, write it next to the line. If the line results in an error, write next it whether it is a compile time error or runtime error, and then proceed as if that line were not there.

```java
public class Person {
    public String name;
    public int age;

    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void greet(Person other) {System.out.println("Hello, " + other.name);}
}


public class Grandma extends Person {

    public Grandma(String name, int age) {
        super(name, age);
    }

    @Override
    public void greet(Person other) {System.out.println("Hello, young whippersnapper");}

    public void greet(Grandma other) {System.out.println("How was bingo, " + other.name + "?");}
}

public class testPeople {
    public static void main(String[] args) {
        Person n = new Person("Neil", 12);
        Person a = new Grandma("Ada", 60);
        Grandma v = new Grandma("Vidya", 80);
        Grandma al = new Person("Alex", 70);
        n.greet(a);
        n.greet(v);
        v.greet(a);
        v.greet((Grandma) a);
        a.greet(n);
        a.greet(v);
        ((Grandma) a).greet(v);
        ((Grandma) n).greet(v);
    }
}
```

2 Re-cursed with Asymptotics!

(a) What is the runtime of the code below in terms of n?

```java
public static int[] curse(int n) {
    if (n <= 0) {
        return 0;
    } else {
        return n + curse(n - 1);
    }
}
```

(b) Assume our BST (Binary Search Tree) below is perfectly bushy. What is the runtime of a single `find` operation in terms of N, the number of nodes in the tree? In this setup, assume a `Tree` has a `Key` (the value of the tree) and then pointers to two other trees, `Left` and `Right`.

```java
public static BST find(BST T, Key sk) {
   if (T == null)
      return null;
   if (sk.compareTo(T.key) == 0))
      return T;
   else if (sk.compareTo(T.key) < 0)
      return find(T.left, sk);
   else
      return find(T.right, sk);
}
```

(c) Can you find a runtime bound for the code below? We can assume the System.arraycopy method takes $\Theta(N)$ time, where N is the number of elements copied. The official signature is `System.arrayCopy(Object sourceArr, int srcPos, Object dest, int destPos, int length)`. Here, `srcPos` and `destPos` are the starting points in the source and destination arrays to start copying and pasting in, respectively, and `length` is the number of elements copied.

```java
public static void silly(int[] arr) {
    if (arr.length <= 1) {
        System.out.println("You won!");
        return;
    }
    int newLen = arr.length / 2
    int[] firstHalf = new int[newLen];
    int[] secondHalf = new int[newLen];
    System.arraycopy(arr, 0, firstHalf, 0, newLen);
    System.arraycopy(arr, newLen, secondHalf, 0, newLen);
    silly(firstHalf);
    silly(secondHalf);
}
```

3 ADT Matchmaking

Match each task to the correct Abstract Data Type for the job by drawing a line connecting matching pairs.

1. You want to keep track of all the unique users who have logged on to your system.	a) List
2. You are creating a version control system and want to associate each file name with a Blob.	b) Map
3. We are running a server and want to service clients in the order they arrive.	c) Set
4. We have a lot of books at our library and we want our website to display them in some sorted order. We have multiple copies of some books and we want each listing to be separate.	d) Queue

4 Disjoint Sets, a.k.a. Union Find

In lecture, we discussed the Disjoint Sets abstruct data structure (ADT). Some authors call this the Union Find ADT. Today, we will use union find terminology so that you have seen both.

(a) Omitted, authored by other

(b) Omitted, authored by other

(c) Omitted, authored by other

(d) What is the runtime for "connect" and "isConnected" operations using our Quick Find, Quick Union, and Weighted Quick Union ADTs? Can you explain why the Weighted Quick union has better runtimes for these operations than the regular Quick Union?

5 A Side of Hashbrowns

We want to map food items to their yumminess. We want to be able to find this information in constant time, so we've decided to use java's built-in HashMap class! Here, the key is an `String` representing the food item and the value is an `int` yumminess rating.

For simplicity, let's say that here a `String`'s hashcode is the first letter's position in the alphabet (A = 0, B = 1... Z = 25). For example, the `String` "hashbrown" starts with "h", and "h" is 7th letter in the alphabet (0 indexed), so the hashCode would be 7. Note that in reality, a `String` has a much more complicated `hashCode()` implementation.

Our hashMap will compute the index as the key's hashcode value modulo the number of buckets in our HashMap. Assume the initial size is 4 buckets, and we double the size our HashMap as soon as the load factor reaches 3/4.

(a) Draw what the HashMap would look like after the following operations.

```java
HashMap<Integer, String> hm = new HashMap<>();
hm.put("Hashbrowns", 7);
hm.put("Dim sum", 10);
hm.put("Escargot", 5);
hm.put("Brown bananas", 1);
hm.put("Burritos", 10);
hm.put("Buffalo wings", 8);
hm.put("Banh mi", 9);
```

(b) Do you see a potential problem here with the behavior of our hashmap? How could we solve this?

6 A Tree Takes on Graphs

Your friend at Stanford has made some statements about graphs, but you believe they are all false. Provide counterexamples to each of the statements below:

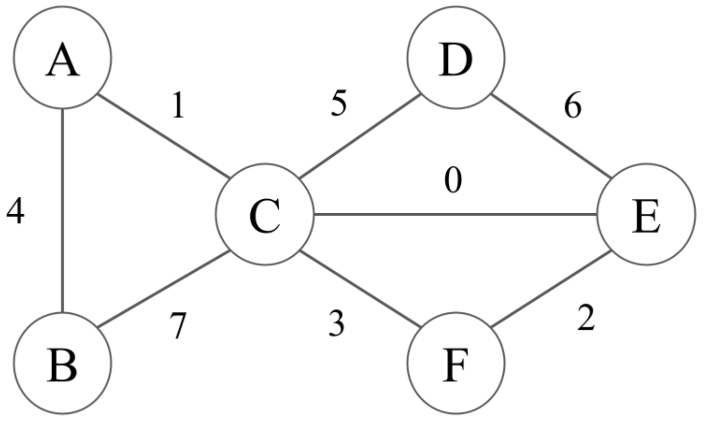
(a) "Every graph has one unique MST."

(b) "No matter what heuristic you use, A* search will always find the correct shortest path."

(c) "If you add a constant factor to each edge in a graph, Dijkstra's algorithm will return the same shortest paths tree."

7 Minimalist Moles

Mindy the mole wants to dig a network of tunnels connecting all of their secret hideouts. There are a set few paths between the secret hideouts that Mindy can choose to possibly include in their tunnel system, shown below. However, some portions of the ground are harder to dig than others, and Mindy wants to do as little work as possible. In the diagram below, the numbers next to the paths correspond to how hard that path is to dig for Mindy.



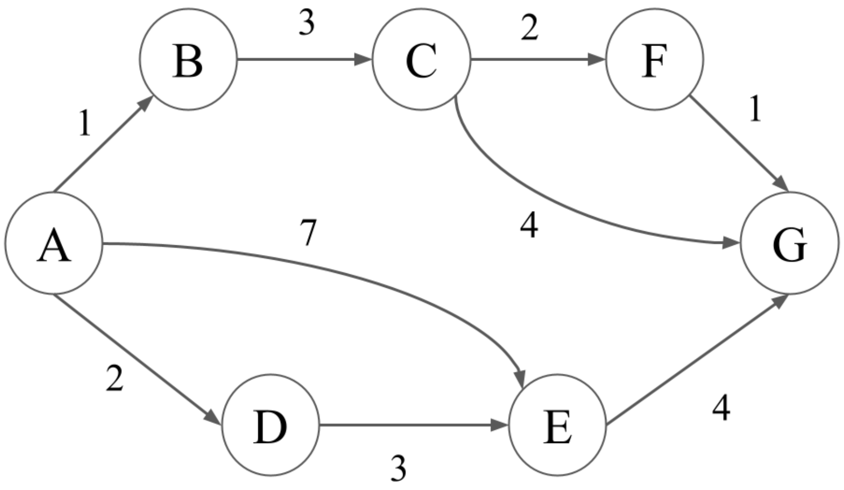
(a) How can Mindy figure out a tunnel system to connect their secret hideouts while doing minimal work?

(b) *Extra:* Find a valid MST for the graph above using Kruskal's algorithm, then Prims. For Prim's algorithm, take A as the start node. In both cases, if there is ever a tie, choose the edge that connects two nodes with lower alphabetical order.

(c) *Extra:* Are the above MSTs different or the same? Is there a different tie-breaking scheme that would change your answer?

8 The Shortest Path To Your Heart

For the graph below, let `g(u, v)` be the weight of the edge between any nodes `u` and `v`. Let `h(u, v)` be the value returned by the heuristic for any nodes `u` and `v`.



Below, the pseudocode for Dijkstra's and A* are both shown for your reference throughout the problem.

Dijkstra's Pseudocode

```
PQ = new PriorityQueue()
PQ.add(A, 0)
PQ.add(v, infinity) # (all nodes except A).

distTo = {} # map
distTo[A] = 0
distTo[v] = infinity # (all nodes except A).

while (not PQ.isEmpty()):
  popNode, popPriority = PQ.pop()

  for child in popNode.children:
    if PQ.contains(child):
      potentialDist = distTo[popNode] +
        edgeWeight(popNode, child)
      if potentialDist < distTo[child]:
        distTo.put(child, potentialDist)
        PQ.changePriority(child, potentialDist)
```

A* Pseudocode

```
PQ = new PriorityQueue()
PQ.add(A, h(A))
PQ.add(v, infinity) # (all nodes except A).

distTo = {} # map
distTo[A] = 0
distTo[v] = infinity # (all nodes except A).

while (not PQ.isEmpty()):
  poppedNode, poppedPriority = PQ.pop()
  if (poppedNode == goal): terminate

    for child in poppedNode.children:
      if PQ.contains(child):
        potentialDist = distTo[poppedNode] +
          edgeWeight(poppedNode, child)

        if potentialDist < distTo[child]:
          distTo.put(child, potentialDist)
          PQ.changePriority(child, potentialDist + h(child))
```

(a) Run Dijkstra's algorithm to find the shortest paths from A to every other vertex. You may find it helpful to keep track of the priority queue. We have provided a table to keep track of best distances, and the edge leading to each vertex on the currently known shortest paths.

	A	B	C	D	E	F	G
DistTo							
EdgeTo							

(b) *Extra:* Given the weights and heuristic values for the graph above, what path would A* search return, starting from A and with G as a goal? Note that the edge weights provided below for your convenience are the same as in the image.

Edge weights	Heuristics
$g(A, B) = 1$	$h(A, G) = 7$
$g(B, C) = 3$	$h(B, G) = 6$
$g(C, F) = 2$	$h(C, G) = 3$
$g(C, G) = 4$	$h(F, G) = 1$
$g(F, G) = 1$	$h(D, G) = 6$
$g(A, D) = 2$	$h(E, G) = 3$
$g(D, E) = 3$	
$g(E, G) = 4$	
$g(A, E) = 7$	

	A	B	C	D	E	F	G
DistTo							
EdgeTo							

9 Sorta Interesting, Right?

(a) What does it mean to sort ”in place”, and why would we want this?

(b) What does it mean for a sort to be ”stable”? Which sorting algorithms that we have seen are stable?

(c) Which algorithm would run the fastest on an already sorted list?

(d) Given any list, what is the ideal pivot for quicksort?

(e) So far, in class, we’ve mostly applied our sorts to lists of numbers. In practice, how would we typically make sure our sorts can be applied to other types?

10 Zero One Two-Step

Note: this question is a popular LeetCode question, sometimes called the "Dutch National Flag" problem.

(a) Given an array that only contains 0's, 1's and 2's, write an algorithm to sort it in linear time. You may want to use the provided helper method, `swap`.

```
public static int[] specialSort(int[] arr) {
    int front = 0;
    int back = arr.length - 1;
    int curr = 0;

}
```

```
private static void swap(int[] arr, int i, int j) {
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

(b) We just wrote a linear time sort, how cool! Can you explain in a sentence or two why we can't always use this sort, even though it has better runtime than Mergesort or Quicksort?